Abstract

A GUI simulator created to illustrate the inner workings of the Memory Management Unit. Each step of the simulation shows the operation performed by the MMU.

Contents

[Abstract 2](#_Toc513886658)

[Introduction 3](#_Toc513886659)

[Basic concepts 3](#_Toc513886660)

[Objectives of the project 3](#_Toc513886661)

[In the following sections 3](#_Toc513886662)

[State of the art 3](#_Toc513886663)

[Methodology 3](#_Toc513886664)

[Evaluation 3](#_Toc513886665)

[Conclusion 3](#_Toc513886666)

[State of the art 4](#_Toc513886667)

[Virtual memory 4](#_Toc513886668)

[Paging 4](#_Toc513886669)

[Memory Management Unit (MMU) 5](#_Toc513886670)

[Page replacement algorithm 5](#_Toc513886671)

[FIFO page replacement algorithm 5](#_Toc513886672)

[Least Recently Used replacement algorithm 5](#_Toc513886673)

[Random replacement algorithm 5](#_Toc513886674)

[Methodology 6](#_Toc513886675)

[Design of the simulator 6](#_Toc513886676)

[Use case diagram 6](#_Toc513886677)

[Class diagram 7](#_Toc513886678)

[Implementation 7](#_Toc513886679)

[Evaluation 8](#_Toc513886680)

[GUI evaluation 8](#_Toc513886681)

[Conclusion 9](#_Toc513886682)

[Further improvements 9](#_Toc513886683)

[References 10](#_Toc513886684)

# Abstract

Virtual Memory is used in each one of our personal computers, most of the users are oblivious of this abstraction, but us, computer scientist must be aware of it if we want to understand its advantages and drawbacks. This knowledge would allow us to write more efficient programs.

Creating a graphical simulator for students will create the opportunity for them to study this abstraction and to experiment with different setups of the page replacement algorithms.

The solution of the project was first to design the model of the system, and then implement it using C#. To create the GUI for the application the WPF was used, to create intuitive UI and allow fast prototyping. Unit tests were written as more features were added to the project.

The results of this little project are a complete GUI app which may be used for educational purposes. The FIFO replacement algorithm was implemented, hence testing of the whole system was possible.

The main takeaway from this project is that the objectives of the project were achieved and that there is room for improvement.

# Introduction

Virtual memory is an abstraction, used to create the illusion of a large main memory, it allows the extension of memory by swapping out pages into the external storage (usually the hard disk).

The subject of this project is creating GUI simulator in order to let students study the inner workings of the paging mechanism.

## Basic concepts

The basic concepts and components which are related to this project are the following:

* MMU: manages the memory, swapping in and swapping out the pages, and maintaining the mappings from the virtual memory to the physical memory.
* Virtual memory: the processes view of the memory, they do not interact with physical memory directly.
* Page: a fixed sized contiguous block of memory used in the paging process.
* Paging: the process of dividing the memory in pages, allowing the swapping of memory contents between the main memory and the external memory.

## Objectives of the project

Create a GUI simulator where users can:

* view the contents of the memory
* view the mappings between virtual pages and physical pages
* allow writing to addresses and read from addresses.
* View the history of operations performed by the MMU

## In the following sections

### State of the art

Theoretical background for the project will be presented in detail, such as the basic concepts introduced earlier.

### Methodology

Design of the program, implementation details, GUI design.

### Evaluation

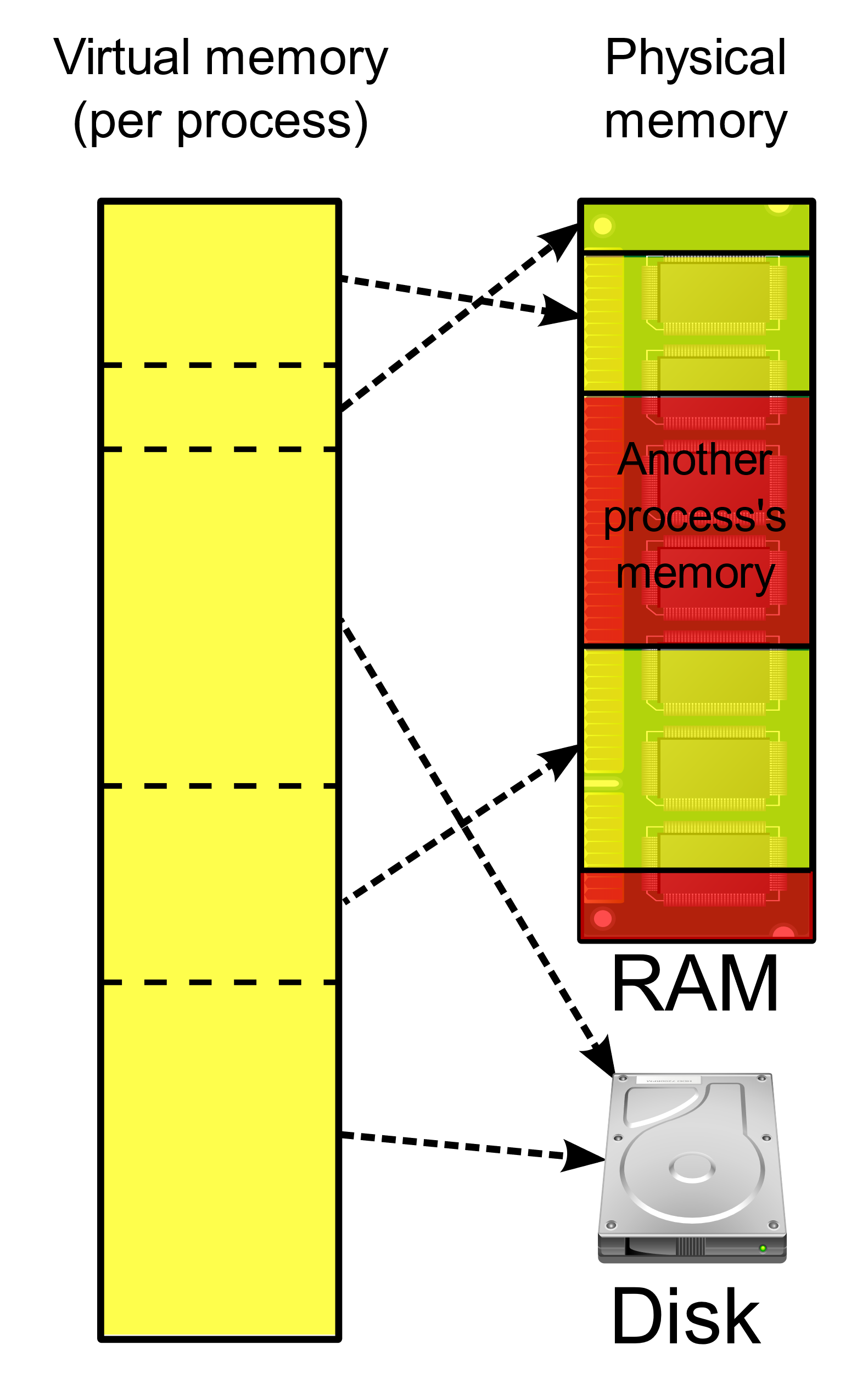
Testing of the program and final result: unit testing and system testing.

### Conclusion

Final thoughts and future improvements.

# State of the art

## Virtual memory

Virtual memory is an abstraction, it creates the illusion of a main memory which is very large. But since we could not create a such a large physical memory, we should create a mapping between the two memories, meaning that some addresses in the virtual memory correspond to some addresses in the RAM (physical memory) and some addresses in the virtual memory will correspond to addresses in the swap memory (on the disk). This way there could be two different types of memory present in the system, with different strengths and weaknesses, but the process will see treat them uniformly, since it only has to deal with the virtual one. 

## Paging

A page is a fixed length contiguous block of memory.

Paging is present when there are two types of memories: the main memory (RAM) and the secondary memory, usually the hard drive. Paging will allow us to retrieve pages from the secondary memory and put it in the main memory, and vice versa. It allows for isolation between processes and

## Memory Management Unit (MMU)

Performs the translation from virtual address space to physical address space.

When paging is present, there is a page table containing so called page table entries, which are basically mappings between the virtual address to the physical address.

Using these entries, the MMU can decide if the page is in main memory, is in the swap or it is not loaded at all. If caching is implemented, then in these page table entries are stored the “dirty bit” which will tell us if the page has been changed and must be written back.

It is the responsibility of the MMU to find a page which will replaced in case a page must be swapped in to the main memory:

* Knowing we need to access a page currently in swap,
* Find a page in the RAM
* Replace found page with page we should access.

These steps occur when a so-called page fault occurred.

## Page replacement algorithm

Finding a page in the RAM is done by an algorithm named page replacement algorithm. The purpose of the algorithm is to find a page which will likely not be accessed by subsequent requests.

Choosing a good page replacement algorithm will increase performance by avoiding unnecessary swaps.

### FIFO page replacement algorithm

Using a simple queue always remove the first page from the queue and put the inserted page at the end of the queue.

### Least Recently Used replacement algorithm

The basic idea is that the page which has been used in the recent instructions is likely to be used again in the future.

Thus, we could replace the page which has been the least recently accessed.

One of the best performing algorithms in practice.

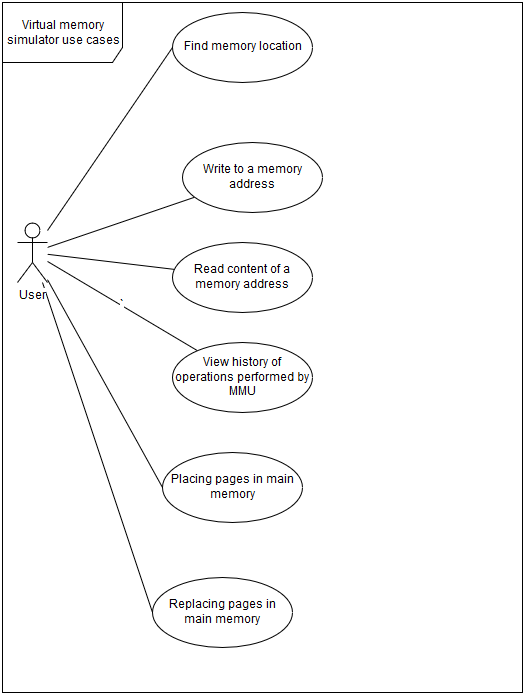
### Random replacement algorithm

Just choose a page at random and replace it. It works surprisingly well, in some cases even as good as LRU replacement algorithm.

# Methodology

## Design of the simulator

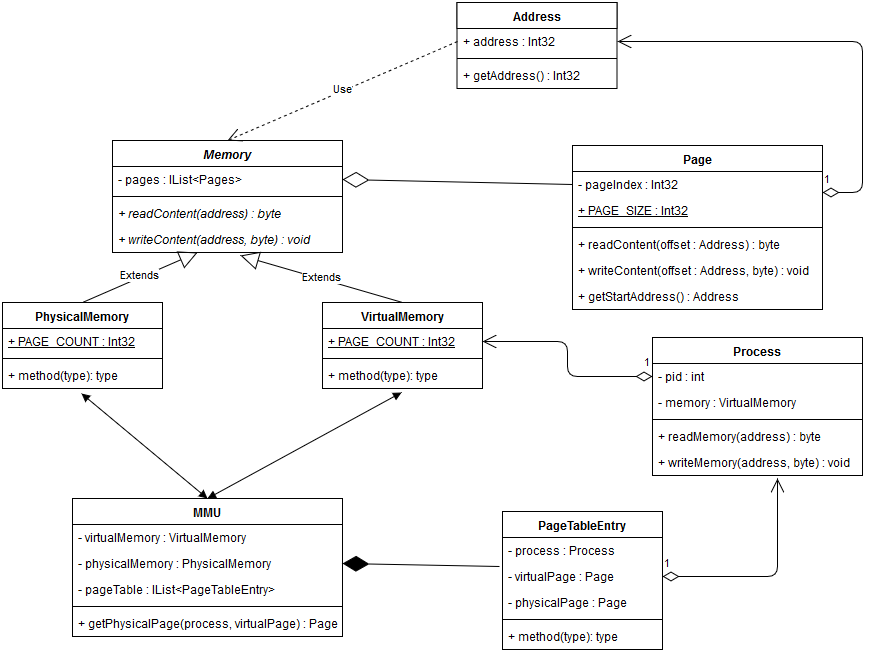
### Use case diagram



I identified some of the use cases which are relevant for the implementation of the simulator.

These use cases were already enumerated/introduced in the preceding section (Introduction).

### Class diagram



This is the initial class diagram of the system, and it can be considered as the main design drawing of the system. It does not include the GUI design, which was created using MVVM pattern (Model-View-ViewModel) supported by WPF.

## Implementation

The simulator was implemented in C#, using the Windows Presentation Foundation (WPF) for the GUI interface.

You may check out the source code if you are interested in the details of the implementation, but it roughly follows the design.

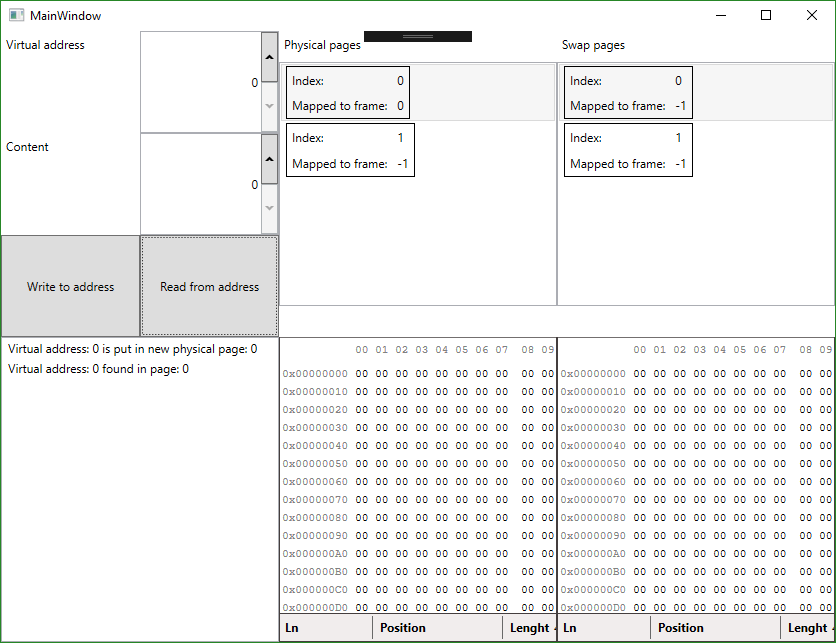
# Evaluation

The model of the program was evaluated using unit tests and integration tests. These tests were separated into another sub project.

The some of the tests are using random numbers to simulate the real-life usage of the program.

System testing was done by hand to test the non-functional requirements of the projects and the seamless integration between the components and the look and feel of the application.

## GUI evaluation



Here you can see the GUI of the app, you be the judge of how good it is looking.

# Conclusion

The project could be considered successful, if we look at the objectives defined at the introduction of this document, since they were all satisfied.

Since new page replacement algorithms could be easily added by sub-classing the MMU class, we can conclude that the project is easily expandable, and should the need arise we may add a new page replacement algorithm to the simulator.

## Further improvements

In addition to adding more algorithms to the system, we could design it in such a way to change these algorithms at run-time and to set the number of pages of the simulation also at run-time, by a user selected value.

Exporting configuration and results to a file could be one of the main additions, with head to head with reading instructions from a file, to avoid typing so much of the commands.   
This would allow creating a testing program, where the users could easily test the different versions of the page replacement algorithms.

# References

General information about virtual memory

<https://en.wikipedia.org/wiki/Virtual_memory>

General information about paging

<https://en.wikipedia.org/wiki/Paging>

General information about Memory Management Unit (MMU)

<https://en.wikipedia.org/wiki/Memory_management_unit>